Nonexistence of Williamson matrices of order 65

P. Graham
https://orcid.org/0009-0007-8493-1505

November 4, 2026

Abstract

We prove by exhaustive computation that Williamson matrices of order 65 do not exist: no four symmetric circulant sign matrices of that order have squares summing to 260 times the identity. Compression to length 13, common multipliers, conservative power spectral density bounds and Williamson's product identity reduce the search to matching pairs of full sequences. The CPU computation took about 271 CPU-hours. Two checkers verify the compressed enumeration: one reuses the search's row generator, and one regenerates rows with separately written code. They share the compression model and per-representative enumeration records; neither repeats the decisive full-row lift search. Full-pipeline controls reproduce known smaller-order counts. The search is computational under stated arithmetic assumptions and is not formally verified.

This work has not yet been reviewed by experts. The proofs, the search code, the enumeration checkers, the witness checkers and the text were produced by AI models (from OpenAI and Anthropic), working under the author's direction.

1 Introduction

A *sign matrix* has entries in $\{1, -1\}$. A matrix is circulant when each row is the cyclic shift of its predecessor. Four symmetric circulant sign matrices A, B, C, D of order n are *Williamson matrices* if

$$A^2 + B^2 + C^2 + D^2 = 4nI_n, \tag{1}$$

where I_n is the identity matrix. We establish the following computer-assisted result.

Theorem 1. *There are no four symmetric circulant sign matrices A, B, C, D of order* 65 *satisfying $A^2 + B^2 + C^2 + D^2 = 260I_{65}$.*

The proof uses the written reductions below and the exhaustive computation described in Section 5. This settles the existence of Williamson matrices of order 65. Kharaghani, Mohammadian, Tayfeh-Rezaie and Zaitsev list 65 and 67 among orders where ordinary Williamson matrices had not been found [16, Section 2.3]. Their order-65 construction is near-Williamson: only three of its blocks must be symmetric [16, Definition 1.1, Table 6]. Bright had identified 65 as the smallest unresolved odd order [15, Section 4.2].

Williamson's construction [1] is described historically by Koukouvinos and Kounias [12, Section 1, p. 162]. Turyn gave an infinite family [2, abstract]. Koukouvinos and Kounias [12, Section 1, p. 162] credit Baumert, Golomb and Hall with order 23 and Baumert and Hall with the classification at odd orders through 23 [3, 4]. Đoković's note [6, Sections 1 and 5, Table 1] lists the searches of Sawade, Koukouvinos and Kounias, and Đoković [10, 11, 12, 5, 7], with the corrected count at 25 and incomplete earlier enumerations at 33 and 39, and nonexistence at 35. Holzmann, Kharaghani

and Tayfeh-Rezaie's classification through 59 gives nonexistence at 47, 53 and 59 [8, Section 4, Table 1]. Egan, Kotsireas, London and Ó Catháin summarize nonexistence at 35, 47, 53 and 59 [17, Section 2.1]. Lang and Schneider searched the Turyn-type subclass through order 99 [9]. Bright, Kotsireas and Ganesh attribute the order-61 enumeration to them [14, Section 2]; the latter paper itself enumerates order 63, finding two equivalence classes, one from Turyn's construction and one new. Thus all odd orders below 65 were settled: the nonexistence orders are $35, 47, 53, 59$, and the others have examples [14, Section 2]. The present result settles the smallest remaining odd order identified by Bright [15, Section 4.2], completing the existence classification through odd order 65. The order-39 nonexistence claim in [12] was incorrect, as witnessed by the later examples and enumeration [7, 8].

Fourier and PSD tests for complementary sequences were used earlier by Fletcher, Gysin and Seberry [21]; compression and its spectral consequences are developed by Đoković and Kotsireas [13]. We reuse Holzmann, Kharaghani and Tayfeh-Rezaie's row and pair spectral bounds, expressed below as power spectral density (PSD) bounds, and product-identity elimination [8, Eq. (2), Theorem 1, algorithm 2(c–d)]. They fix two blocks, eliminate the fourth and solve linearized equations modulo 4 for the third before checking the original equations [8, Section 3, Eqs. (4–5), steps 2(d)(ii)(A–E)]. Here Đoković–Kotsireas compression [13, Definition 3, Theorem 3], in Bright–Kotsireas–Ganesh's formulation [14, Definition 11, Theorem 12], is followed by a multiplier restriction on one sequence and two-sided PSD bounds before matching pairs of full sequences. Bright, Kotsireas and Ganesh give the spectral formulation in Corollaries 8–9 [14, Corollaries 8–9]. Our contribution is the complete order-65 enumeration. The compressed conditions are necessary; their admissible tuples still require a full-row search. Exhausting those lifts is the hard step.

The proof translates the matrix equation into autocorrelations, then exhausts full-row lifts of compressed representatives. The reductions preserve every solution; the final autocorrelation comparison is exact.

Nearby work includes Kharaghani, Mohammadian, Tayfeh-Rezaie and Zaitsev's near-Williamson matrices [16, Definition 1.1, Table 6] and Bennett, Bright, Colinot and Nayak's enumeration allowing nonsymmetric blocks [18, abstract]. Zhang's account of an AI research system reports a prediction that Williamson matrices of order 65 are "very likely non-existent", based on reported computational and structural evidence [19, §7.4]. The account explicitly distinguishes this assessment from a formal proof and states that exhaustive search or a SAT+CAS nonexistence proof would be required for mathematical certainty [19, §7.5]. The exhaustive computation here confirms that prediction; no code or reductions from Zhang's account are used. Kishan independently agrees [20, §101].

2 Rows, autocorrelations, and spectra

The row sum and power spectral density give the necessary conditions for individual sequences.

Lowercase a, b, c, d denote the four first rows. For the equivalence counts below, two quadruples are equivalent under the relation generated by row negations, permutations of the four rows, and a common invertible multiplier of the cyclic indices.

All indices are taken modulo the odd order n, and $H = (n-1)/2$. For example, at order 5 a normalized symmetric first row has the form $(1, \varepsilon, \eta, \eta, \varepsilon)$, with $\varepsilon, \eta \in \{1, -1\}$. In general a normalized row is a sequence $x = (x_i)_{i \in \mathbb{Z}/n\mathbb{Z}}$ with $x_0 = 1$ and $x_i = x_{-i}$. Define its periodic autocorrelation by

$$\mathrm{PAF}_x(s) = \sum_{i \in \mathbb{Z}/n\mathbb{Z}} x_i x_{i+s}.$$

For a symmetric circulant matrix with first row x, the entry in position $(0, s)$ of its square is $\mathrm{PAF}_x(s)$. Thus the matrix equation is exactly

$$\mathrm{PAF}_a(s) + \mathrm{PAF}_b(s) + \mathrm{PAF}_c(s) + \mathrm{PAF}_d(s) = \begin{cases} 4n, & s = 0, \\ 0, & s \neq 0. \end{cases} \tag{2}$$

Whenever a sum below runs over the four rows, it counts their four positions, including repetitions. Autocorrelation is symmetric in s, so testing $0 \leq s \leq H$ suffices.

A symmetric row of order 65, normalized to start with $+1$, is encoded by its 32 remaining signs. For a sequence x, its negative support is the set of indices where $x_i = -1$; its weight is the number of such indices. The following encoding records one index from each nonzero symmetric pair. Encode it by the H-bit integer m_x whose bit $i-1$ is one exactly when $x_i = -1$, for $1 \leq i \leq H$. The function $\mathrm{popcount}(m_x)$ counts the one bits of this mask.

Lemma 2. *If Williamson matrices of odd order n exist, their first rows may be normalized to $x_0 = 1$. Each normalized row has row sum $r_x = n - 4\,\mathrm{popcount}(m_x)$, and the four row sums satisfy $r_a^2 + r_b^2 + r_c^2 + r_d^2 = 4n$.*

Proof. Negation preserves every autocorrelation. Negative entries in a normalized symmetric row occur in pairs, giving the stated row sum. Summing (2) over all shifts gives the sum of the squares of the row sums. □

At order 65 the nondecreasing lists of absolute row sums are exactly

$$(1, 3, 5, 15),\ (1, 3, 9, 13),\ (3, 3, 11, 11),\ (3, 7, 9, 11),\ (7, 7, 9, 9).$$

Each sign is forced by $r_x \equiv 65 \pmod 4$, giving the allowed signed values $1, -3, 5, -7, 9, -11, 13, -15$. Permuting the four rows therefore permits sorting by the absolute row sums without loss.

For frequency $k \in \mathbb{Z}/n\mathbb{Z}$, the real eigenvalue of the circulant is

$$\lambda_x(k) = 1 + 2\sum_{i=1}^{H} x_i \cos(2\pi i k/n).$$

The Fourier transform diagonalizes all circulant matrices. Consequently (2) is equivalent to

$$\lambda_a(k)^2 + \lambda_b(k)^2 + \lambda_c(k)^2 + \lambda_d(k)^2 = 4n \qquad (k \in \mathbb{Z}/n\mathbb{Z}). \tag{3}$$

For symmetric sequences the power spectral density is $\lambda_x(k)^2$. In particular every solution row satisfies $\lambda_x(k)^2 \leq 4n$. The enumeration considers every mask and retains only permitted row sums and rows satisfying this necessary spectral bound, with the numerical slack justified in Section 4.1.

3 Compression and multiplier orbits

Compression and common multipliers reduce the number of quadruples to be searched. We describe the compressed classes and their symmetry reduction.

In this section capitals denote compressed sequences, while lowercase letters continue to denote full rows.

For $n = qL$, with compression factor q, define the compression of a row x by

$$X_j = \sum_{t=0}^{q-1} x_{j+Lt}, \qquad j \in \mathbb{Z}/L\mathbb{Z}.$$

Symmetry gives $X_{-j} = X_j$, and $\sum_j X_j = \sum_i x_i$. Here $q = 5$ and $L = 13$. Write $\mathrm{PAF}_X(s) = \sum_{j \in \mathbb{Z}/L\mathbb{Z}} X_j X_{j+s}$ for compressed autocorrelation. Write A, B, C, D for the compressions of a, b, c, d, respectively.

Lemma 3. *For four rows satisfying (2), their compressions satisfy*

$$\mathrm{PAF}_A(s) + \mathrm{PAF}_B(s) + \mathrm{PAF}_C(s) + \mathrm{PAF}_D(s) = \begin{cases} 4n, & s \equiv 0 \pmod{L}, \\ 0, & s \not\equiv 0 \pmod{L}. \end{cases}$$

Proof of Lemma 3. Expanding the compressed autocorrelation gives $\mathrm{PAF}_X(s) = \sum_{t=0}^{q-1} \mathrm{PAF}_x(s + Lt)$. For a nonzero residue s none of these shifts is zero modulo n. For $s = 0$ exactly one is zero. Summing over the four rows proves the claim. □

A *multiplier* is an invertible residue u modulo 65, acting on a row by $x_i \mapsto x_{ui}$. It preserves (2), since it permutes the shifts. Its action on the compression depends only on its residue modulo 13: the compression of the transformed row at j is $\sum_{t=0}^{4} x_{uj+13ut} = X_{uj \bmod 13}$, since $\gcd(u, 5) = 1$ makes ut run through all residues modulo 5. Opposite multipliers act identically on symmetric rows. Every invertible residue modulo 13 lifts to an invertible residue modulo 65: among $u, u + 13, \ldots, u + 52$, exactly one is divisible by 5, so the other four are coprime to 65.

A class consists of the retained rows with the same compressed vector; its signed row sum is already the sum of that vector.

Call a tuple of four classes *admissible* when its compressed vectors satisfy Lemma 3 and its row sums satisfy Lemma 2. A *lift* is a tuple of four retained rows, one from each class; it need not itself solve the matrix equation. These are compression classes, rather than multiplier orbits; lifting recovers the uncompressed sequences. The enumeration producing compressed tuples is called the generator; the separately written enumeration checking completeness is called the cover check. Both use the following ordering convention. Classes are ordered lexicographically by the seven entries $(X_0, \ldots, X_6)$ of their symmetric compression. A class identifier is its zero-based rank in this order; the data files use those identifiers. A *sorted* class tuple has nondecreasing absolute row sums and, within equal row sums, nondecreasing class identifiers. Acting by a multiplier means applying it to each compressed vector and then re-sorting within equal-row-sum positions. The generator and cover check use this same order.

The admissible class tuples are enumerated by matching sums of two exact compressed autocorrelation vectors to the complementary pair sums. Every ordered class pair is included; equal classes are permitted. Sorting positions of equal absolute row sum removes only row permutations. The complete tuple set is also enumerated by separate code matching the exact integer vectors themselves.

Lemma 4. *Let one representative of each orbit of admissible sorted class tuples under* $(\mathbb{Z}/13\mathbb{Z})^*/\{1, -1\}$ *be retained. Every order-*65 *Williamson solution is equivalent, by a common multiplier and a permutation of its rows, to a lift of a retained tuple.*

Proof. A solution's tuple is admissible by Lemma 3. Choose a multiplier carrying it to its orbit representative, lift that multiplier modulo 65, and apply it to all four full rows. Re-sort rows of equal absolute row sum. The resulting solution has the representative tuple. All lifts of the representative are searched, even if more than one multiplier carries the original tuple to it. Multipliers permute the exact eigenvalues. Transformed solution rows therefore still satisfy $\lambda_x(k)^2 \le 260$ and pass the numerical filter with its proved margin. □

The compressed orbit representatives are checked for invalid, duplicate or missing entries by separately enumerating the exact compressed tuples.

A further symmetry reduces one row list in each orbit representative. The subgroup

$$K = \{u \in (\mathbb{Z}/65\mathbb{Z})^* : u \equiv 1 \pmod{13}\} = \{1, 14, 27, 53\}$$

fixes every compressed vector. Choose one designated position in the representative and retain only rows whose mask is least in its K-orbit. Applying a common element of K that minimizes that row preserves a solution in the same representative. It is essential that this restriction applies to only one designated row; independent minimization of all four rows would not be justified.

4 Pair filtering and autocorrelation matching

The PSD test and product identity retain every solution when pairs of sequences are matched. A pairing is one of the three splits $(a, b) \mid (c, d)$, $(a, c) \mid (b, d)$, or $(a, d) \mid (b, c)$. For each representative the computation minimizes the sum of the two pair counts over these three pairings and the four choices of designated row, using the list sizes after that row's multiplier restriction. Ties take the first choice in the displayed order and row order a, b, c, d. The displayed splits specify the left and right sides.

For a fixed representative, its four classes supply four lists of full rows. Pairs are grouped by a fixed key function of their autocorrelation sums and masks; on the opposite side we use the negative sums and complementary masks. Equal data therefore give equal keys. A pair survives when it satisfies the PSD tests. A key match is a surviving left pair and a surviving right pair with equal keys. Appendix C supplies the exact hashing and storage rules.

4.1 Conservative numerical enclosures

We use integer enclosures to make the spectral comparisons safe under rounding. At order 65, the error in each computed squared eigenvalue is below $4 \cdot 10^{-11}$ under the arithmetic assumptions and cosine-table checks detailed in Appendix B. The row filter rejects only when the computed square exceeds $260 + 10^{-6}$; it therefore retains every solution row.

Let $\widetilde{\lambda}_x(k)$ be the computed eigenvalue. Define integer lower and upper enclosures

$$\ell_x(k) = \max\{0, \lfloor 64\widetilde{\lambda}_x(k)^2 - 0.01 \rfloor\}, \qquad h_x(k) = \lceil 64\widetilde{\lambda}_x(k)^2 + 0.01 \rceil.$$

The error bound, including the two additional floating-point additions, implies

$$\ell_x(k) \le 64\lambda_x(k)^2 \le h_x(k). \tag{4}$$

Clamping the lower bound to zero preserves this enclosure, since a square is nonnegative.

4.2 The pair filter

The opposite pair of row lists bounds the possible PSD contribution of each pair. Consider two lists $\mathcal{A}, \mathcal{B}$ on one side and $\mathcal{C}, \mathcal{D}$ on the other, after restricting the designated row. Set $Q = 64 \cdot 260$. For each frequency let

$$U(k) = \max\{0, Q - \min_{c \in \mathcal{C}} \ell_c(k) - \min_{d \in \mathcal{D}} \ell_d(k)\},$$

$$V(k) = \max\{0, Q - \max_{c \in \mathcal{C}} h_c(k) - \max_{d \in \mathcal{D}} h_d(k)\}.$$

Retain a pair (a, b) only when, at every frequency,

$$\ell_a(k) + \ell_b(k) \le U(k), \qquad h_a(k) + h_b(k) \ge V(k). \tag{5}$$

The opposite side uses the same rule with the two sides interchanged.

Lemma 5. *Every pair belonging to a solution in the four lists passes (5) on both sides.*

Proof. For its complementary rows c, d, equation (3) gives $64(\lambda_a(k)^2 + \lambda_b(k)^2) = Q - 64\lambda_c(k)^2 - 64\lambda_d(k)^2$. Replacing each square by the appropriate lower or upper enclosure proves the inequalities before clamping. Raising a negative upper bound to zero only weakens the upper test. Raising a negative lower bound to zero changes nothing, because $h_a + h_b$ is nonnegative. □

If a list is empty there is no solution in the representative. The implementation details are given in Appendix C.

4.3 Williamson's product identity

The second matching condition is Williamson's classical product identity [1]; see the modern statement [14, Theorem 14]. We include its short argument to fix the normalization.

In set notation the negative positions form supports; their weights give the row sums, and symmetric difference corresponds to exclusive-or in the stored encoding. We retain that encoding in the following statement. In its proof a sequence also denotes the corresponding element of the cyclic group ring.

Lemma 6. *For an odd-order Williamson solution normalized by $x_0 = 1$ in each row, $a_i b_i c_i d_i = -1$ for every nonzero index i. Equivalently, $m_a \oplus m_b \oplus m_c \oplus m_d = 2^H - 1$, where $\oplus$ is bitwise exclusive-or on H bits.*

Proof. In the integral group ring of the cyclic group G of order n, write J for the sum of all group elements and T_x for the sum of the positions where row x is negative. Symmetry means x is fixed by the involution $g \mapsto g^{-1}$, so $x^2 = xx^{(-1)} = \sum_{s \in G} \mathrm{PAF}_x(s) s$. Then $x = J - 2T_x$ and $x^2 = nJ - 4|T_x|J + 4T_x^2$. Symmetry and normalization make $|T_x|$ even. Substituting in $\sum_x x^2 = 4n\,1$ and dividing by 4 in $\mathbb{Z}[G]$ gives

$$nJ - \Big(\sum_x |T_x|\Big) J + \sum_x T_x^2 = n\,1.$$

Reducing modulo 2, with n odd and each $|T_x|$ even, gives $\sum_x T_x^2 \equiv 1 + J$. Modulo 2, each T_x^2 equals the sum of the squares of its individual group elements, because mixed terms occur twice. Thus

$$\sum_x \sum_{t \in T_x} t^2 = 1 + J \pmod 2.$$

Squaring permutes a cyclic group of odd order. Every nonidentity position therefore occurs in an odd number of the four negative sets, proving the product identity and its mask form. □

4.4 Proof of the main theorem

The preceding reductions preserve every solution and reduce the exclusion to the finite computation summarized in Table 1.

Proof of Theorem 1. Negating a row preserves its autocorrelations, so every solution can be normalized. Lemma 2 and the PSD bound retain its four rows. A common invertible index change permutes autocorrelation shifts, preserving (2). Applying such a change to all four rows and permuting them carries their compressions to a retained representative (Lemmas 3 and 4). Choose $u \in K$ minimizing the designated row's mask. Since K fixes every compressed vector, the class tuple is unchanged. Applying u to all rows preserves the solution; only that position is restricted.

For $1 \le s \le H$, equation (2) gives $\mathrm{PAF}_a(s) + \mathrm{PAF}_b(s) = -(\mathrm{PAF}_c(s) + \mathrm{PAF}_d(s))$. The conservative PSD tests retain these pairs (Lemma 5), and their product condition holds by Lemma 6. For the fixed linear hash $f(x) = \sum_{s=1}^{H} R_s \, \mathrm{PAF}_x(s)$ modulo 2^{64}, these equalities give $f(a) + f(b) = -f(c) - f(d)$ and $m_a \oplus m_b = (2^H - 1) \oplus m_c \oplus m_d$. Both sides apply the same key function to these two values. A solution therefore produces equal keys. Collisions can only add candidates, each of which is tested by the exact autocorrelations in (2). The exhaustive computation finds none (Table 1), completing the exclusion. □

5 Computational verification

The exhaustive computation is supported by checks of the compressed enumeration and validation against smaller orders. The complete CPU search found no matching pairs (Table 1). The proof depends on exhaustive computation, the written reductions above, and correct compiler, hardware, integer and IEEE binary64 arithmetic, including the interval checks in Appendix B; the search has not been formally verified and the records of the enumeration do not constitute a short proof certificate. One checker reuses the search's row generator with separate compressed-enumeration code; the other has separately written row, cosine, enumeration and record-checking code. Both share the compression model, ordering conventions and per-representative records of list sizes, pair counts, survivor counts and matches. They check completeness of the compressed enumeration and the recorded totals, without independently rederiving all survivor or key-match counts or recomputing every pair comparison. A separately written NumPy filter-and-join implementation exactly reproduces list sizes and both survivor counts for 312 order-65 representatives (300 random and the 12 largest by pair count) and 301 at order 55. At order-65 representative 5200, a 32-bit-key positive join test exercises key matching and exact verification; independent NumPy pair regeneration recounts matches and confirms zero exact solutions. These checks do not replace rerunning the full search, which takes about 271 CPU-hours. Its implementation and numerical details are in Appendices C and B, with commands in the accompanying checker documentation.

Quantity	Count
Normalized sequences examined	2^{32}
Retained sequences	396,050,997
Compressed classes	15,251
Admissible sorted class tuples	423,527
Representatives after multiplier reduction	70,699
Enumerated pairs, both sides	70,830,746,132,488
Surviving pairs, both sides	1,645,891,558,124
Matches between pair keys	0
Solutions	0

Table 1: Counts in the complete CPU enumeration at order 65.

The archived unfiltered baseline enumeration, separate from the full-row search, reproduces the published numbers of equivalence classes through odd order 33 [8, Table 1]; the counts are listed in Appendix A.

All controls described here use the programs distributed with this paper. Full-pipeline controls at 25, 35, 45 and 55, all with compression factor 5, reproduce respectively 10, 0, 1 and 1 equivalence classes [8, Table 1]. They exhaust all 43, 195, 3,309 and 9,409 compressed representatives. Their solution closures under the equivalences above contain 1,500, 0, 144 and 240 normalized ordered quadruples. At 25 and 35, the reference is the separate unfiltered baseline. At 45 and 55, the closures of the witnesses discovered by our search contain the witnesses printed in [8, Table 3]; each has one equivalence class, matching [8, Table 1]. The reference quadruples are read only after the search, never supplied as seeds. These controls exercise row generation, compression, multiplier reduction, the designated-row restriction, both PSD bounds and the hash join.

The programs also recover known solutions without supplied witnesses at 39 with compression factors $3, 13$, at 45 with factors $3, 5, 9, 15$, and at 55 with factors $5, 11$ [8, Section 4, Table 1]. A separate NumPy implementation agrees on all 3,309 compressed representatives at order 45 and recovers its solution.

The CPU computation took about 271 CPU-hours; its processor models and full elapsed time were not recorded.

The search programs, input data and checker documentation accompany the paper. The mathematical programs and data are unchanged from those archived at https://doi.org/10.5281/zenodo.23622210.

Appendix A Enumeration totals

The CPU search regenerates its inputs from the problem definition; Table 1 summarizes the resulting enumeration. Pair counts sum both sides for each representative, counting their occurrences separately. The complete enumeration has no unresolved quadruple of classes.

The smaller-order enumeration reported in Section 5 gives the following counts:

n	3	5	7	9	11	13	15	17
classes	1	1	2	3	1	4	4	4
n	19	21	23	25	27	29	31	33
classes	6	7	1	10	6	1	2	5

Appendix B Numerical error bound

Conservative error bounds justify the PSD comparisons of Section 4.1.

The row generator and the search evaluate the cosine table in IEEE binary64 arithmetic, rounded to nearest, without fast-math transformations. The numerical bound assumes ordinary IEEE binary64 operations rather than x87 extended intermediate precision; a build changing the arithmetic model requires its own justification. The actual table entries are checked against interval evaluations of cosine at 50 decimal digits. Their errors are at most 2^{-49}. For order 65 the eigenvalue sum is split into two sums of at most 16 terms. Each partial sum has magnitude at most 16, so the error in one half is bounded by

$$16\,2^{-49} + 16^2\,2^{-53} < 6 \cdot 10^{-14}.$$

Combining the halves, doubling, and adding one gives eigenvalue error below $2.6 \cdot 10^{-13}$. Since $|\lambda_x(k)| \le 65$, the error in the computed square, including rounding of the multiplication, is below $4 \cdot 10^{-11}$. Fusing a multiplication and addition does not enlarge this bound. The row filter rejects only when the computed square exceeds $260 + 10^{-6}$, so it retains every solution row.

Appendix C Hashing and storage

The storage scheme preserves every pair equality required by a solution. The fixed coefficients and key function are specified in the accompanying implementation documentation. Their values affect collisions and cost, but the argument needs only that both sides use the same function.

C.1 Hash matching and exact verification

We show that the matching keys retain every exact autocorrelation equality.

For each row, form a linear hash of its nonzero-shift integer autocorrelations,

$$f(x) = \sum_{s=1}^{H} R_s \,\mathrm{PAF}_x(s) \pmod{2^{64}},$$

using fixed odd coefficients R_s; their reproducible choice is described in the accompanying implementation documentation. A solution satisfies $f(a) + f(b) = -f(c) - f(d)$ and $m_a \oplus m_b = (2^H - 1) \oplus m_c \oplus m_d$. The search combines these two necessary equalities into a single 64-bit key, using the same function on both sides. Specifically it uses $\alpha \oplus R\beta$ for hash sum α and mask value β, with a fixed odd R, and maps a zero key to one. The right side uses the negative hash sum

and the H-bit complement of its mask XOR. Equal inputs give equal keys; hash collisions may add candidates but cannot remove a solution.

Moreover, $f(a) + f(b) + f(c) + f(d) = \sum_{s=1}^{H} R_s(\mathrm{PAF}_a(s) + \mathrm{PAF}_b(s) + \mathrm{PAF}_c(s) + \mathrm{PAF}_d(s)) = 0$ modulo 2^{64}, so their join keys are equal.

The join retains every pair with an equal key, including duplicates. Its storage and restart rules, justified in Section C.2, do not remove any such pair.

Finally, each key match is checked using the four exact integer autocorrelation vectors at shifts $0, \ldots, H$. Equation (2) then decides the matrix equation exactly. A separate witness checker can also form the four matrices and multiply them directly. This last check prevents false positives; it cannot repair a missing pair earlier in the search.

C.2 Storage

The storage rules preserve all surviving pairs and their exact comparisons.

The pair bounds are formed in signed integer arithmetic before conversion to unsigned 16-bit values. The stored row values are below 16,642; all intervening sums and differences fit the signed integer type. Frequency reordering and vectorized comparisons do not change the predicates. In a vectorized upper test, a row with $\ell_x > U$ is discarded first, before forming $U - \ell_x$; the lower difference is $\max\{0, V - h_x\}$. The remaining frequencies are tested with signed integer sums. Empty lists contain no pairs and cannot yield a solution.

Every surviving pair is retained, including repeated keys and repeated sequences. Looking up a key examines all pairs with that key. If storage is insufficient, the whole attempt restarts with pairs partitioned by the high bits of the same key on both sides; equal keys therefore remain together. Every reported order-65 representative completed without such partitioning. The accompanying implementation documentation specifies the storage scheme.

References

[1] J. Williamson, Hadamard's determinant theorem and the sum of four squares, Duke Mathematical Journal **11** (1944), 65–81. https://doi.org/10.1215/S0012-7094-44-01108-7.

[2] R. J. Turyn, An infinite class of Williamson matrices, Journal of Combinatorial Theory, Series A **12** (1972), 319–321. https://doi.org/10.1016/0097-3165(72)90095-7.

[3] L. Baumert, S. W. Golomb and M. Hall, Jr., Discovery of an Hadamard matrix of order 92, Bulletin of the American Mathematical Society **68** (1962), 237–238. https://doi.org/10.1090/S0002-9904-1962-10761-7.

[4] L. Baumert and M. Hall, Jr., Hadamard matrices of the Williamson type, Mathematics of Computation **19** (1965), 442–447. https://doi.org/10.1090/S0025-5718-1965-0179093-2.

[5] D. Ž. Đoković, Williamson matrices of orders $4 \cdot 29$ and $4 \cdot 31$, Journal of Combinatorial Theory, Series A **59** (1992), 309–311. https://doi.org/10.1016/0097-3165(92)90071-2.

[6] D. Ž. Đoković, Note on Williamson matrices of orders 25 and 37, Journal of Combinatorial Mathematics and Combinatorial Computing **18** (1995), 171–175. https://combinatorialpress.com/jcmcc-articles/volume-018/note-on-williamson-matrices-of-orders-25-and-37/.

[7] D. Ž. Đoković, Williamson matrices of order $4n$ for $n = 33, 35, 39$, Discrete Mathematics **115** (1993), 267–271. https://doi.org/10.1016/0012-365X(93)90495-F.

[8] W. H. Holzmann, H. Kharaghani and B. Tayfeh-Rezaie, Williamson matrices up to order 59, Designs, Codes and Cryptography **46** (2008), 343–352. Cited version: author manuscript dated December 12, 2007 (page and algorithm numbering refer to this version). https://web.archive.org/web/20170810183932id_/http://math.ipm.ac.ir/~tayfeh-r/papersandpreprints/Williamson.pdf. https://doi.org/10.1007/s10623-007-9163-5.

[9] W. Lang and E. Schneider, Turyn type Williamson matrices up to order 99, Designs, Codes and Cryptography **62** (2012), 79–84; first published online March 5, 2011. https://doi.org/10.1007/s10623-011-9492-2.

[10] K. Sawade, Hadamard matrices of order 100 and 108, Bulletin of the Nagoya Institute of Technology **29** (1977), 147–153. https://cir.nii.ac.jp/crid/1520572358172744832.

[11] C. Koukouvinos and S. Kounias, Hadamard matrices of the Williamson type of order $4 \cdot m$, $m = p \cdot q$ an exhaustive search for $m = 33$, Discrete Mathematics **68** (1988), 45–57. https://doi.org/10.1016/0012-365X(88)90040-4.

[12] C. Koukouvinos and S. Kounias, There are no circulant symmetric Williamson matrices of order 39, Journal of Combinatorial Mathematics and Combinatorial Computing **7** (1990), 161–169. https://combinatorialpress.com/jcmcc-articles/volume-007/there-are-no-circulant-symmetric-williamson-matrices-of-order-39/.

[13] D. Ž. Đoković and I. S. Kotsireas, Compression of periodic complementary sequences and applications, arXiv:1302.0571v1, February 4, 2013; Designs, Codes and Cryptography **74** (2015), 365–377. https://arxiv.org/abs/1302.0571v1. https://doi.org/10.1007/s10623-013-9862-z.

[14] C. Bright, I. Kotsireas and V. Ganesh, Applying Computer Algebra Systems with SAT Solvers to the Williamson Conjecture, arXiv:1804.01172v2, May 14, 2019; first posted April 3, 2018; Journal of Symbolic Computation **100** (2020), 187–209. https://arxiv.org/abs/1804.01172v2. https://doi.org/10.1016/j.jsc.2019.07.024.

[15] C. Bright, Extended Research Statement, March 20, 2019. https://cs.curtisbright.com/extended-research-statement.pdf.

[16] H. Kharaghani, A. Mohammadian, B. Tayfeh-Rezaie and V. Zaitsev, A Search for Hadamard Matrices of Near-Williamson Type, arXiv:2605.08661v2, September 6, 2026. First posted May 9, 2026 as version 1 by H. Kharaghani, A. Mohammadian and B. Tayfeh-Rezaie, titled *A search for Hadamard matrices of Williamson type*. https://arxiv.org/html/2605.08661v2.

[17] R. Egan, I. Kotsireas, S. London and P. Ó Catháin, Recent progress in the construction of Hadamard matrices, arXiv:2610.27175v1, first posted October 19, 2026. https://arxiv.org/abs/2610.27175v1.

[18] A. Bennett, C. Bright, P. Colinot and A. Nayak, Quaternionic Perfect Sequences and Hadamard Matrices, arXiv:2601.22337v1, first posted January 29, 2026. https://arxiv.org/abs/2601.22337v1.

[19] S. Y. Zhang, Programmable Emergence via Cognitive Engine: Longitudinal Evidence of AI Research Maturation, Zenodo record 19392279, version posted April 2, 2026. https://doi.org/10.5281/zenodo.19392279.

[20] Kishan, New Hadamard matrices, GitHub repository, file `search/NOTES.md`, Section 101, "Williamson and good matrices of order 65", committed October 25, 2026; reports by an exhaustive search of all compressed instances that no Williamson matrices of order 65 exist. https://github.com/vatskishan03/Hadamard/blob/809480f37d6baf39b2e0689da5027a66705a36af/search/NOTES.md.

[21] R. J. Fletcher, M. Gysin and J. Seberry, *Application of the discrete Fourier transform to the search for generalised Legendre pairs and Hadamard matrices*, Australasian Journal of Combinatorics **23** (2001), 75–86. https://ajc.maths.uq.edu.au/pdf/23/ocr-ajc-v23-p75.pdf.